PROPIEDADES NOS CONTEDORES GRID

Para comezar a maquetar empregando GRID o primeiro que temos que facer é definir cal das nosas etiquetas HTML se vai a convertir no **contedor GRID**. Unha vez o temos decidido darémoslle unha destas propiedades:

- **display:grid** se queremos que a nosa reixa (o noso grid) sexa un elemento de bloque.
- **display:inline-grid** se queremos que o noso grid sexa un elemento en liña.

Co obxecto de que os exemplos se amosen de maneira máis clara, empregaremos a primeira de elas.

Unha vez temos asignado esta propiedade ao contedor, todos os elementos que contén pasan a convertirse de maneira automática en elementos do GRID cuxa colocación e propiedades poderemos comezar a modificar dende o contedor.

Definición da estrutura do GRID

Normalmente o primeiro paso que daremos para maquetar con GRID é definir a estrutura que vai a ter a nosa reixa. É un paso importante e para evitar problemas despois é convinte realizar unha reflexión sobre elo.

Unha vez decidimos que estrutura queremos, empregaremos unha serie de propiedades para definila. As máis importantes para comezar son:

- **grid-template-columns:** Para definir o número e tamaño das diferentes columnas da nosa estrutura. Debemos de poñer tantos valores de anchura como columnas queiramos que teña o GRID.
- **Grid-template-rows:** Para definir o número e tamaño das diferentes filas da miña estrutura. Debemos de poñer tantos valores de altura como filas queiramos que teña o GRID.
- **Grid-row-gap:** Para establecer a separación entre as diferentes columnas.
- **Grid-column-gap:** Para establecer a separación entre as diferentes filas.

Nos dous últimos simplemente estamos expresando distancias pero os dous primeiros teñen moitas posibilidades así que vamos a amosar varios exemplos:

Exemplos con columnas:

```
/* Tres columnas que se reparten o 100% do contedor*/
grid-template-columns: 20% 50% 30%;

/* Catro columnas. Tres de tamaño fixo 100px e a outra ocupa o espazo libre restante */
grid-template-columns: 100px auto 100px 100px;

/* Catro columnas. Todas cun tamaño igual */
grid-template-columns: auto auto auto auto;

/* Tres columnas cada unha con nome (entre []). Dúas con tamaño fixo e a outra ocupando o
espazo restante */
grid-template-colums: [id] 100px [nome] 300px [apelidos] auto;
```

Exemplos con filas:

```
    /* Tres filas que se reparten toda a altura do contedor (a que sexa) */
    grid-template-rows: 20% 50% 30%;

    /* Catro filas. Tres de altura fixa 100px e a outra ocupará o resto do espazo libre
ata encher todo o contedor en altura.*/
```

```
    grid-template-rows: 100px auto 100px 100px;

    /* Catro filas que se reparten de maneira equitativa o alto do contedor */
    grid-template-rows auto auto auto auto;

    /* Tres filas (todas con nome, entre corchetes) Dúas delas con tamaño fixo e a
restante ocupará todo o alto libre. */
    grid-template-rows: [uno] 100px [dos] 300px [tres] auto;
```

Nestas dúas propiedades tamén podemos repetir valores e usar a unidade **fr** que me serve para establecer ratios para que os elementos se repartan o espazo restante. Podemos velo mellor cun par de exemplos.

```
/* Catro columnas. Tres de 20% con nome col-start. A última ocupará o resto do espazo
libre */
grid-template-columns: repeat(3, 20% [col-start]) auto;
/* Catro columnas. Unha de tamaño fixo e as demáis repártense o espazo libre en 5 partes
da seguinte maneira (2+1+2) */
grid-template-columns: 2fr 100px 1fr 2rf;
```

Aínda que pode parecer complexo é sinxelo e con moi poucas liñas podemos conseguir estruturas complexas. Para velo imos a poñer un exemplo:

Supoñemos que temos o seguinte HTML:

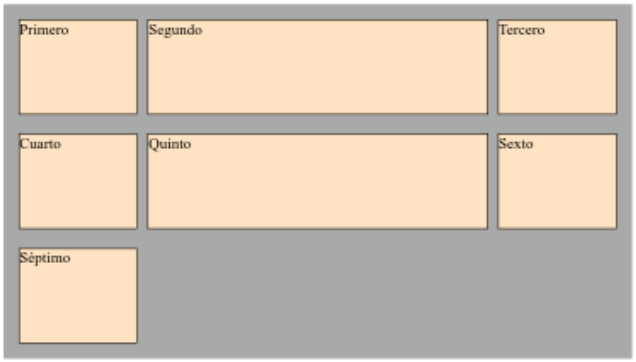
```
<div class="container">
  <div>Primeiro</div>
  <div>Segundo</div>
  <div>Terceiro</div>
  <div>Carto</div>
  <div>Quinto</div>
  <div>Sexto</div>
  <div>Sétimo</div>
</div>
```

Empregando só o seguinte CSS:

```
.container {
  background-color: #aaa;
  display: grid;
  grid-column-gap: 10px;
  grid-row-gap: 20px;
  grid-template-columns: 20% auto 20%;
  grid-template-rows: repeat(3, 100px);
  margin: 20px auto;
  padding: 1em;
  width: 80%;
}

.container > div {
  background-color: bisque;
  border: 1px solid black;
}
```

Obteño a seguinte estrutura:



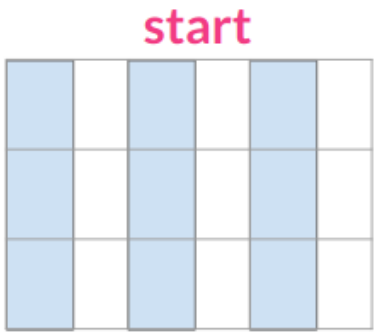
Vemos como se distribuíron os 7 elementos nunha cuadrícula, nun grid de 3x3 seguindo a estrutura que lle dixemos.

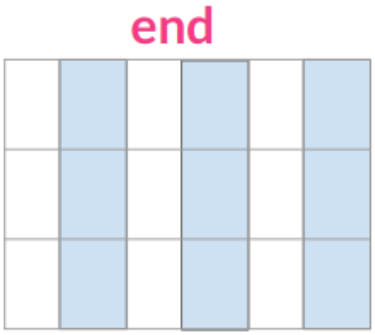
Aliñación Horizontal

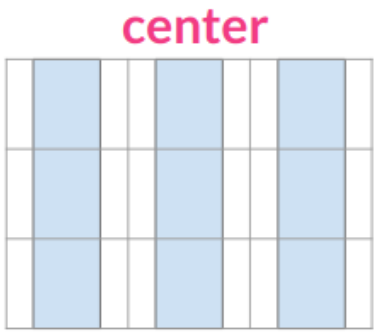
Por defecto os elementos do GRID ocupan todo o ancho da cela que lle corresponde pero podemos optar por outro tipo de aliñacións horizontais dando valores á propiedade **justify-items**. Os diferentes valores que pode tomar son os seguintes:

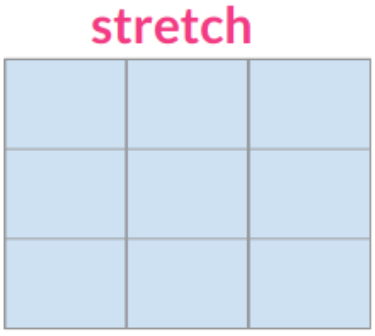
- **start**
- **end**
- **center**
- **stretch** que é a opcion por defecto.

Entenderase mellor que fai cada un mediante unha explicación visual:







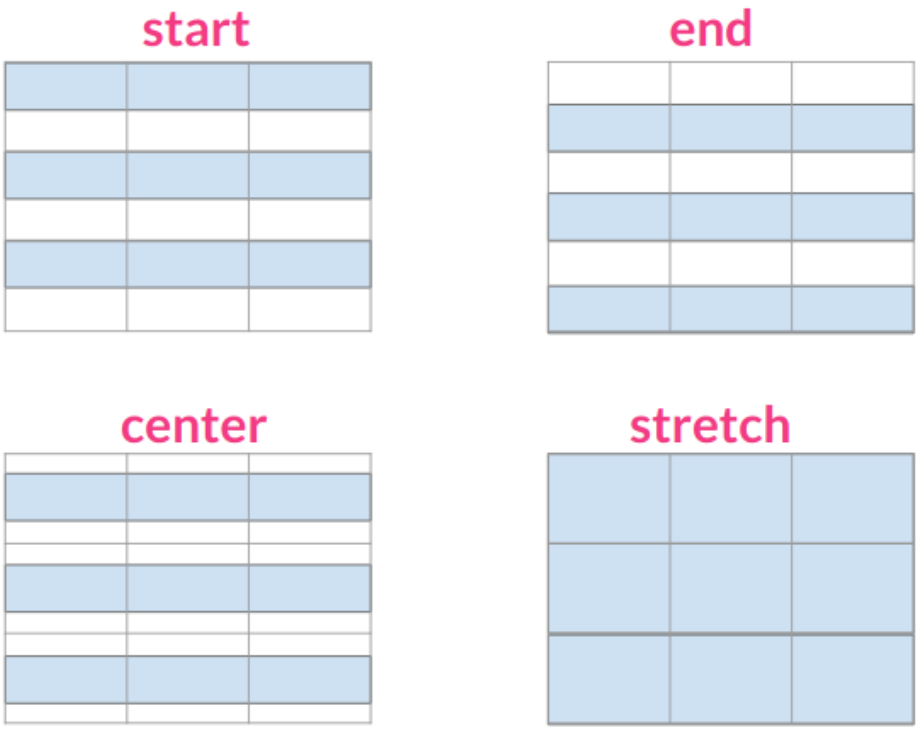


Aliñación Vertical

Moi similar ao anterior. Por defecto os elementos do GRID ocupan todo o alto de la cela que lle corresponde pero podemos optar por outro tipo de aliñacións verticais dando valores á propiedade **align-items**. Los diferentes valores que puede tomar son los siguientes:

- **start**
- **end**
- **center**
- **stretch** que é a opción por defecto.

Entenderase mellor que fai cada un mediante unha explicación visual:

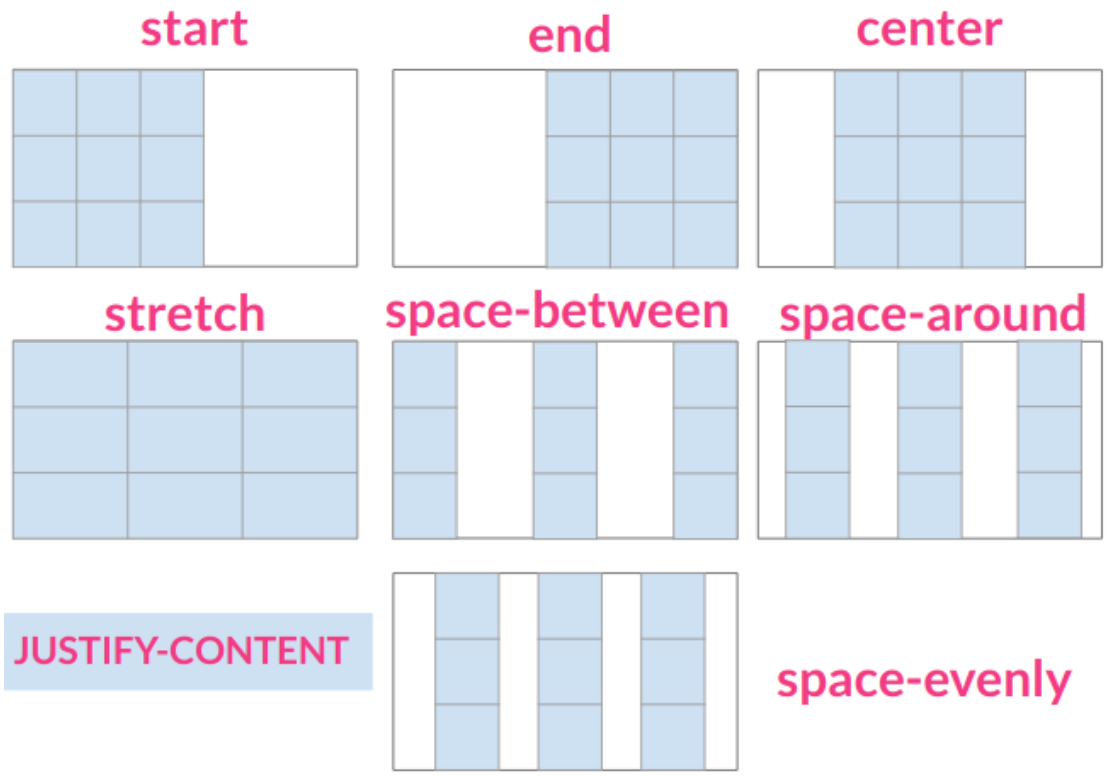


Se quero xuntar estas dúas últimas aliñacións empregarei a propiedade **place-items** indicando primeiro o valor para **align-items** e despois o valor para **justify-items**.

Distribución dentro do contenedor

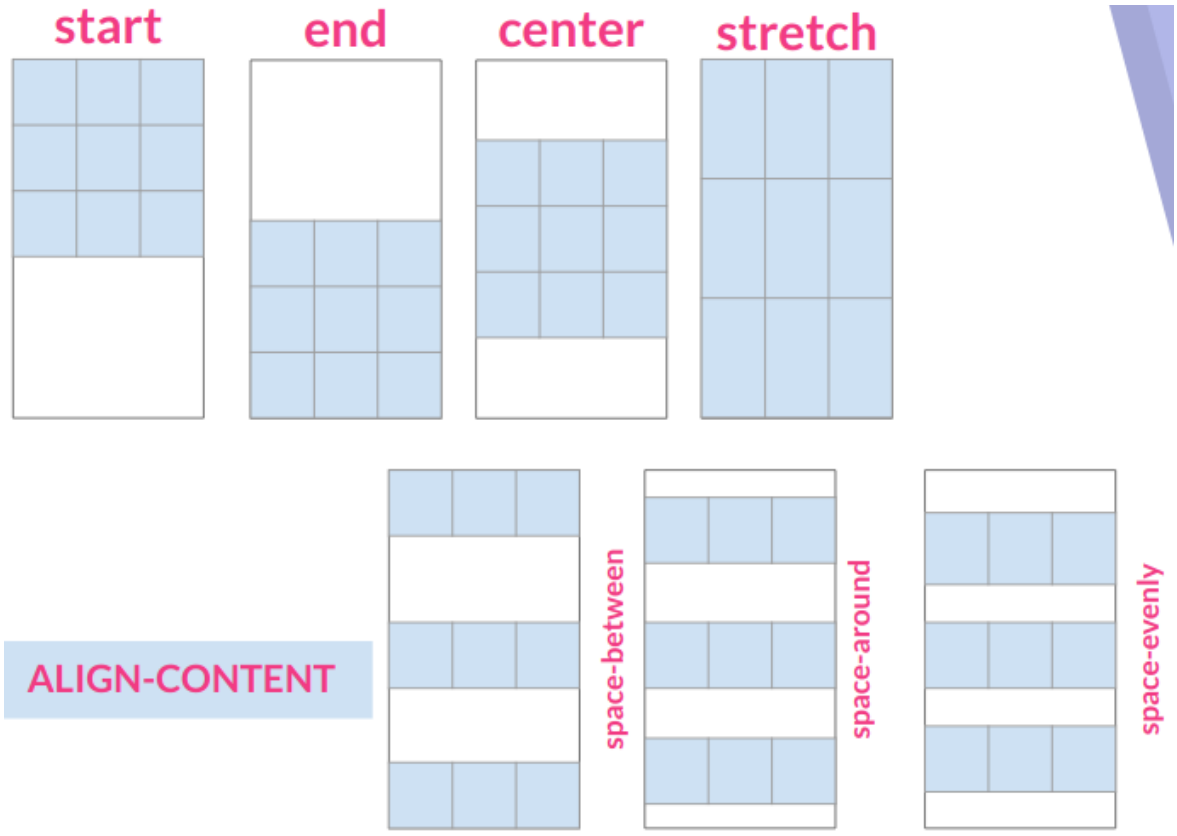
En determinados casos pode suceder que os elementos do GRID non ocupen todo o ancho ou todo o alto do contedor GRID. Nestas ocasións podo distribuir as columnas e as filas empregando as propiedades **justify-content** (horizontal) y **align-content** (vertical). Ambas poden tomar os mesmos valores e para entender mellor esos valores e como funcionan imos a presentar dúas imaxes.

Para a propiedade **justify-content**:



A zona azul representa as columnas que están dentro dun rectángulo que é o contedor GRID.

Para a propiedade **align-content**:



A zona azul representa as columnas que están dentro dun rectángulo que é o contedor GRID.

Se quero xuntar estas dúas últimas aliñacións empregarei a propiedade **place-content** indicando primeiro o valor para **align-content** e despois o valor para **justify-content**.